# Neural Networks and other Techniques for Fault Identification and Isolation of Aircraft Systems

**M. Innocenti[1] and M. Napolitano[2]**

[1]Department of Electrical Systems and Automation
University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy

[2]Department of Mechanical and Aerospace Engineering
G-60 Engineering Science Building, West Virginia University
Morgantown, WV 26506, USA

## Summary

Fault identification, isolation, and accomodation have become critical issues in the overall performance of advanced aircraft systems. Neural Networks have shown to be a very attractive alternative to classic adaptation methods for identification and control of non-linear dynamic systems. The purpose of this paper is to show the improvements in neural network applications achievable through the use of learning algorithms more efficient than the classic Back-Propagation, and through the implementation of the neural schemes in parallel hardware. The results of the analysis of a scheme for Sensor Failure, Detection, Identification and Accommodation (SFDIA) using experimental flight data of a research aircraft model are presented. Conventional approaches to the problem are based on observers and Kalman Filters while more recent methods are based on neural approximators. The work described in this paper is based on the use of neural networks (NNs) as on-line learning non-linear approximators. The performances of two different neural architectures were compared. The first architecture is based on a Multi Layer Perceptron (MLP) NN trained with the Extended Back Propagation algorithm (EBPA). The second architecture is based on a Radial Basis Function (RBF) NN trained with the Extended-MRAN (EMRAN) algorithms. In addition, alternative methods for communications links fault detection and accomodation are presented, relative to multiple unmanned aircraft applications.

## Introduction

The previous decade has witnessed a dramatic increase in neural networks (NN) research, mostly induced by the introduction of the Back-Propagation algorithm for feedforward NNs with supervised learning. During the initial stage of this growth NNs have been applied to pattern recognition and classification, which can be classified as static problems. Within a later trend NNs were proposed for identification and control of dynamic systems of both linear and nonlinear nature. The reason behind this application lies in the fact that, despite progresses were made in adaptive control theory for linear systems, a solution for the identification and control of a non-linear uncertain system was yet to be formulated. A complete review on NN theoretical principles is widely available in the literature.

In general, a neural network is defined as an architecture featuring input and output parameters interconnected through one or more layers of neurons, also called processing elements, which provide a non-linear activation. The operations at each neuron consist of adding the total weighted sum of all the input signals of the preceding layer, with the option of adding a threshold value; this total sum becomes then the argument of a non-linear function, which provides the activation, to generate an output signal. During learning, the weights and the thresholds of the architecture are updated, typically with a gradient based method, with the goal of minimizing a cost function associated with an identification, control or other task.

Fault tolerant flight control systems (FTFCS) are required to accommodate failures in different components, mainly sensors and actuators. Some of the procedures described in this paper focus on the sensor failure problem. Sensor failures are critical when the measurements from the failed sensor are used in the feedback loop of the flight control laws. In fact, sensor failures, if left undetected and uncompensated, can lead to

*Paper presented at the RTO AVT Course on "Intelligent Systems for Aeronautics", held in Rhode-Saint-Genèse, Belgium, 13-17 May 2002, and published in RTO-EN-022.*

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **01 JUN 2003** | **N/A** | **-** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Neural Networks and other Techniques for Fault Identification and Isolation of Aircraft Systems** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Electrical Systems and Automation University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release, distribution unlimited**

13. SUPPLEMENTARY NOTES
**See also ADM001519. RTO-EN-022**

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | |
| **unclassified** | **unclassified** | **unclassified** | **UU** | **26** | |

closed-loop instability and, potentially, unrecoverable flight conditions. Most of today's high performance military aircraft as well as commercial jetliners feature a flight control system with triple physical redundancy in the sensory capabilities. A Fault Detection (FD) scheme monitors these components and, as a sensor failure is detected, it recovers the nominal functionality by switching from the faulty unit to one of the two remaining back-up units. A voting-scheme between the different signals from the sensors measuring the same parameter is typically used for this task. However, there are special purpose aircraft (e.g. UAVs) and spacecraft where reduced complexity, lower costs, and weight optimization are major design specifications.

For these classes of aircraft and spacecraft, an alternative approach can take advantage of the analytical redundancy existing in the system to provide fault tolerance capabilities. Analytical redundancy is the known functional relationship existing between the system's outputs, states and inputs. In other words, the information provided by a set of sensors along with a priori knowledge of the system allows detecting and identifying the faulty sensor, while estimating the related variable as a function of other measured variables. Research on fault tolerance based on analytical redundancy has produced a quite mature framework especially for linear systems. Currently, substantial research challenges are in the extension of the previous schemes to the case of nonlinear systems. Adaptive and on-line approximation methodologies are receiving considerable attention and the use of different types of Neural Networks has been proposed to take advantage of some of their properties. The performance of a NN in approximating a given function critically depends on its underlying structure as well as on the featured training algorithm. The two classes of NNs here considered for sensor failure evaluation are the Multi-Layer Perceptron NNs (MLP NNs) and the Radial Basis Function NNs (RBF NNs). Due to their approximation and adaptation capabilities, both MLP and RBF NNs have been proposed as estimators to approximate complex non-linear systems. Within a larger picture both classes of NNs can be considered as "nonlinear approximators" and a systematic procedure based on Lyapunov's theory for creating non-linear approximators with stability characteristics could be applied to each class.

Other aspects of interest in fault detection are those related to succesfull missions of unmanned air vehicles, possibly flying in formation. Here, one of the main concerns is the reliability of the overall communications system, since each vehicle relies heavily on it, in order to maintain the desired position within a formation, and follow some prescribed flight path. In this case, there are many forms of analytical redundancy that can take place, based on neural network techniques, or other methodologies using traditional optimizers. One interesting aspect is the capability of using NNs to estimate the wake effects in the absence of position sensor information, or as a redundancy for a sensor information failure. Another application is the formation management flow change due to generic failures in the transmitters and/or receivers. In this case, in addition to fault accomodation, the problem is also that of reconfiguring the overall formation structure, in order to continue the mission and/or the assigned task(s). After a generic fault, a fast reconfiguration procedure is run to restore formation-keeping as quickly as possible. When the formation communications are in a safe configuration again, it may be necessary to switch aircraft positions or move an aircraft to an empty node to maximize the formation keeping and safety of all aircraft inside the formation. Since the node-changing decision is decentralized, the algorithm that makes the decision must be deterministic to avoid simultaneous conflicting decisions by more than one aircraft in response to the same post-fault-reconfiguration requirements. The second problem to solve is distribution of fault event information so that all aircraft can apply appropriate reconfiguration decisions. Even if the available communication channels, those used to exchange flight data for formation control, are sufficient for optimal channel reconfiguration, it is possible that the best aircraft candidate for making the node change decision may not be informed of the fault event.

## Basic Concepts in Identification and Control

A generic multi-input/multi-output (MIMO) discrete-time system with n states, m inputs and l measurable outputs can be represented by:

$$x(k+1) = \Phi\left[x(k), u(k)\right]$$
$$y(k) = \Psi\left[x(k)\right]$$

where: $x(k) = [x_1(k), x_2(k), \ldots, x_n(k)]^T$, $u(k) = [u_1(k), u_2(k), \ldots, u_m(k)]^T$, $y(k) = [y_1(k), y_2(k), \ldots, y_l(k)]^T$.

If the system is linear and time-invariant then the equations take on the well-known form:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = C[x(k)]$$

where $A_{nxn}$, $B_{nxm}$, and $C_{nxl}$ are the traditional state, control and output matrices. The relative system can be then defined through the set of matrices $(A,B,C)$. If the system is linear and known, controllability of the system can be asserted and different feedback control strategies, either of classic compensation type or from modern control theory, can be introduced to meet different specifications. For the identification task at linear conditions, that is if the system is linear and $(A,B,C)$ are partially or completely unknown, the problem of identifying $(A,B,C)$ can be clearly formulated and solved.

However, if the system is non-linear and $\Phi$, $\Psi$ are unknown, the identification problem is quite more complicated. Furthermore, for the control task at non-linear conditions, there is a lack of control schemes regardless whether $\Phi$, $\Psi$ are known or unknown. Adaptive control of uncertain systems is feasible in the linear case. In particular, within this direction, SISO adaptive systems tracking a desired dynamics have been extensively analyzed with the Model Reference Adaptive Control (MRAC) method being one of the most representative approaches.

Let us for example consider the problem of identification and control of SISO nonlinear unknown systems. There are four discrete-time models, which can be used to describe this type of systems:

$$Model\,I: y(k+1) = f[y(k),..,y(k-n)] + \sum_{i=0}^{m} b_i u(k-i)$$
$$Model\,II: y(k+1) = f[y(k),...,y(k-n)] + \sum_{i=0}^{m} b_i u(k-i)$$
$$Model\,III: y(k+1) = f[y(k),...,y(k-n)] + g[u(k),...,u(k-m)]$$
$$Model\,IV: y(k+1) = f[y(k),...,y(k-n),u(k),...,u(k-m)]$$

There has been a large variety of studies related to observability, controllability for these systems as well as for designing controllers and observers for them. In this example we will refer to multi-layer feed-forward NNs with supervised learning performed with gradient based algorithms. The suitability of a NN approach to the general identification and control task becomes somewhat immediate if one considers the following properties.

Applicability to non-linear systems. The applicability of NNs to non-linear systems originates from their mapping capabilities. It has been shown that a feedforward NN with at least one hidden layer is capable of approximating any nonlinear function once enough training is provided. This property can be extremely useful when the input/output data are related to a non-linear time-varying system.

Parallel processing and hardware implementation. NNs have an inherent parallel architecture, which, leads naturally to parallel hardware implementations. These implementation have also the advantages of having, in general, high degree of fault-tolerance and high processing speed, due to the simplicity of the computations (additions, subtractions, multiplications). These advantages have been magnified by the introduction of digital microprocessors with ever increasing performance.

Multivariable systems. NNs, by definition, are multi-input, multi-output (MIMO) entities and this, naturally, leads to their application to multivariable systems.

Learning and adaptation. NNs can be trained using past recorded data or simulated data (off-line training) or current data (on-line learning). Furthermore, among the learning capabilities of a NN, one must differentiate between local and global learning capabilities. Local learning leads to adaptation capabilities to new dynamic conditions while global learning leads to memorization capabilities from previously presented data.

Consider a generic SISO non-linear unknown system described by any of the four models in the previous section. The objective is to provide a NN-based identification scheme for the functions *'f'* and/or *'g'*. A necessary assumption for a well-posed approach is that the system have a bounded response for any bounded set of input data (that is, it is a BIBO stable system). This, in turn, implies that the neural identifier will also need to be BIBO stable. Extensive research efforts are undergoing at this time for the analysis and the proof of stability properties of multi-layer neural networks. Another assumption is that the neural identifier will have to be of series-parallel type meaning that the actual system output $y(k)$, rather than its estimate, is used as

input by the identification scheme during and after the training. The alternative is to use a parallel-type identifier, meaning that the estimate of *y(k)* from the neural identifier is obtained using as input estimates of *y(k)* at previous time steps. It should be underlined that the stability for this type of identifiers has yet to be proved even for simple linear systems. The selection of a series-parallel mode instead of a parallel model is mainly due to the fact that, since the system is assumed to be BIBO and stable, all the input to the NNs are bounded. However, once the neural identifier has been sufficiently trained and the estimation error has converged to a sufficiently low asymptotic value, the series-parallel mode may be replaced by the parallel-working mode. With these working assumptions in place, the next issue to be addressed is related to the selection of the training algorithm for the neural identifiers. To date, the majority of the training for neural networks is being performed with the Back-Propagation (BP) algorithm, a gradient-based optimization method. Slow learning, especially for large order systems, and local minima are "classical" well documented problems of the BP. Unfortunately, these two problems are known to be mutually related and interactive. For example, it has been shown that the learning speed can be improved by setting a proper learning rate. However, this also increases the possibility for the BP to be trapped in a local minimum or to oscillate around the global minimum.

To solve address problems, several alternatives to the BP have been proposed by many researchers. Most of the work has been concentrated on introducing different activation functions or particular procedures for selecting the initial weights. Within this stream of investigations, the authors have used an approach based on the introduction of an heterogeneous network. In an heterogeneous network each neuron in the hidden and output layers has its own capability of updating some new parameters giving the overall architecture increased mapping and adaptation performance. Specifically, in a heterogeneous network each neuron is able to change its output range (upper and lower bounds: U, L) and the slope of the sigmoid activation function (temperature: T). The new non-linear activation function will be given by:

$$f(x_{ij}, U_{ij}, L_{ij}, T_{ij}) = \frac{U_{ij} - L_{ij}}{1 + e^{-x_{ij}/T_{ij}}} + L_{ij}$$

where *i,j* are the indices for the generic neurons of the hidden and output layers and *'x'* is the argument of the classic sigmoid function of the BPA. This function is implemented at each processing element of the hidden layer(s) and output layer. The gradient is calculated to perform the steepest descent as in the BP. The only difference is that the gradient descent will be found with respect to each of the independent variables *'x','U','L','T'*, as opposed to *'x'* only. The training algorithm associated with this heterogeneous network is named Extended Back-Propagation (EBP) and is described with details in the listed references.

The design of on-line learning neural controllers, and neural identifiers can be complicated by the large number of degrees of freedom (number of hidden layers, number of neurons per hidden layer, size of input data window, magnitude of the learning rate and momentum coefficients) in the selection of the neural architectures. Previous work provided the following guidelines for the selection of suitable neural architectures :
- on-line learning neural controllers and neural identifiers are generally very robust to different number of neurons per hidden layer and a different input data window;
- medium-high learning rates, either constant or slowly decreasing with time, performed better for on-line learning neural controllers while low learning rates perform better for on-line learning neural identifiers;
- simple single hidden layer architectures perform as well or better than more complicated and more computationally intensive multiple hidden layers architectures;
- a proper selection of the data to be presented as input and an appropriate choice of the performance indices to be minimized are, by far, the most effective degrees of freedom in the selection of neural architectures.

Next, neural identifiers will be introduced using both BP and EBP algorithms. Consider a different SISO unknown non-linear system described by :

$$y(k+1) = \frac{y(k)}{1 + [y(k)]^2} + [u(k)]^3$$

representing a Model III system. In this case a neural network based identification scheme will consist of two neural identifiers, that is an identifier $N_f$ for the function:

$$f[y(k)] = \frac{y(k)}{1+[y(k)]^2}$$

and an identifier $N_g$ for the function:

$$g[u(k)] = [u(k)]^3$$

such that the estimation dynamics can be expressed as:

$$\hat{y}(k+1) = N_f[y(k)] + N_g[u(k)]$$

As two hidden layer architecture was selected: both neural identifiers are of the type $N_{1,20,10,1}$. The study involved two different phases, that is an initial training phase followed by a testing phase. During the training phase two sets of two similar $N_{1,20,10,1}$ architectures (one using the BP and the other using the EBP) were trained for 100,000 steps with *u(k)* being a random number uniformly distributed between [-2,2] and using a 0.25 learning rate. The range for *u(k)* implies that the estimate of *g* will span between [-8,8] while the estimate of *f* will span between [-10,10]. During this phase the identifier is working in a series-parallel mode, that is the actual system outputs at previous time steps are used by the identifier to provide an estimate at the present time step.

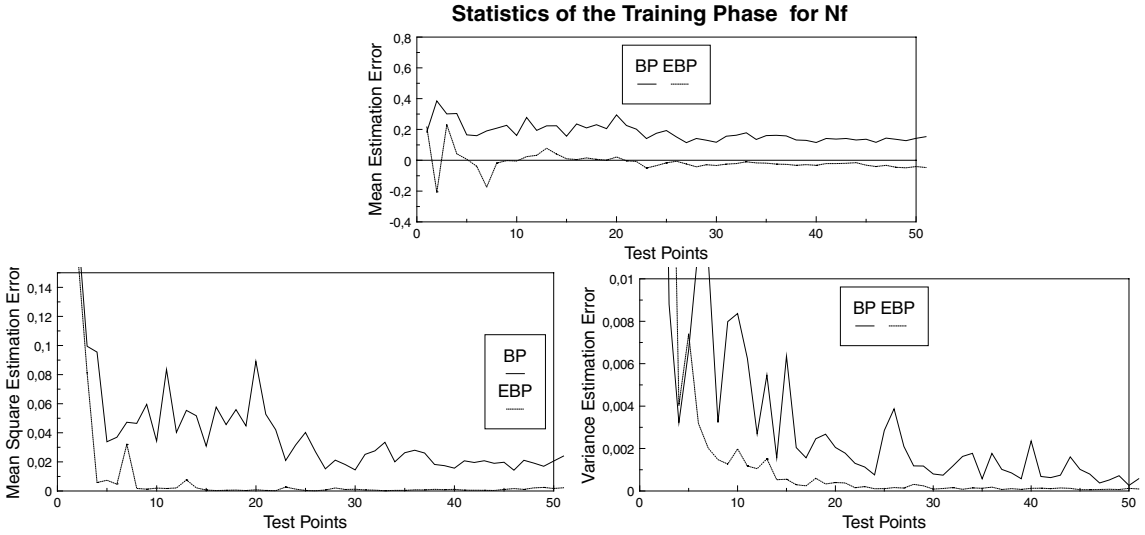


Figure 1. Statistics of Training Phase for Function *'f'*

The training performances associated with the two algorithms were evaluated every 2,000 steps during the 100,000 steps simulation for a total of 50 test points. At each test point the performance of the neural identifier were analyzed using a 100 steps simulation. Classical statistical parameters, that is mean, mean square and variance, were introduced to monitor the estimation errors for *'f'* and *'g'* and given by

$$M_{EE} = \sum_{i=1}^{100} \frac{[\hat{y}(i) - y(i)]}{100} = \sum_{i=1}^{100} \frac{e(i)}{100}, , \; MS_{EE} = \sum_{i=1}^{100} \frac{[\hat{y}(i) - y(i)]^2}{100} = \sum_{i=1}^{100} \frac{e(i)^2}{100} \; V_{EE} = \sum_{i=1}^{100} \frac{[e(i) - M_{EE}]^2}{100}$$

In terms of complexity and required computational effort for the neural architectures, the number of parameters to be updated at each time is 354 using the EBP and 261 using the BP. Therefore, there is a 35% increase in complexity using the EBP due to the additional parameters *U,L,* and *T*. The statistical results are shown in Figure 1 for the identification of the function *'f'*, similar results are available for function *'g'*. The

trends for $M_{EE}$, $MS_{EE}$ and $V_{EE}$ show that the EBP clearly outperforms the BP. In particular, it can be seen that the estimates using BP exhibit a bias most likely caused by the BP algorithm getting stuck in a local minimum point. Efforts were made in repeating the training phase with the BP several times; however, local minimum points problems seemed always to be present to a certain extent. The bias problem for the BP is even more clear in Figure 2, showing estimates of *'f'* following the training phase. After the 100,000 steps training phase the two combined neural identifiers, working in a parallel mode, providing an estimate for *y(k+1)* are tested for 1000 steps with the control sequence $u(k)=sin(2\pi k/25)+sin(2\pi k/10)$. The results are reported in Figure 3 and Table I indicating better performance of the EBP algorithm. The reason for the improved performance of the EBP vs. the BP algorithm has to be searched in the different learning characteristics.

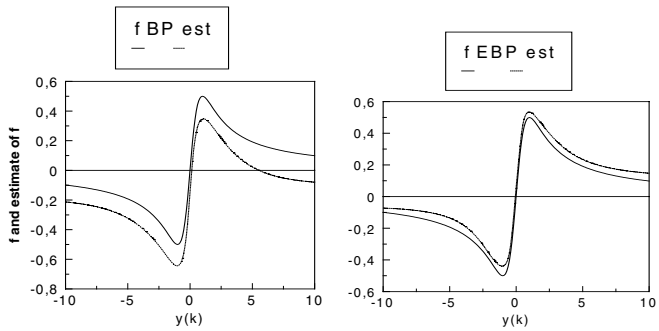


Figure 2. Testing Phase, Estimate of Function *'f'*

Table I. Statistics of Estimation Error: Testing Phase

| | BP Algorithm | EBP Algorithm |
|---|---|---|
| Time | 100,001-100,100 | 100,001-100,100 |
| Mee | -.0277 | -.0154 |
| Msee | .0017 | .000518 |
| Vee | .000945 | .000750 |

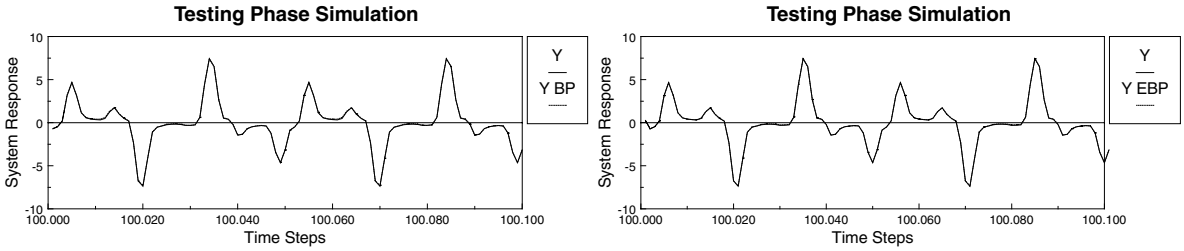


Figure 3. Testing Phase: Simulation

The reason for the improved performance of the EBP vs. the BP algorithm has to be searched in the different learning characteristics. The EBP has more degrees of freedom in the learning through the use in the gradient method of the following partial derivatives :

$$\frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial x_{ij}}=-\frac{\left(O_{ij}-U_{ij}\right)\left(O_{ij}-L_{ij}\right)}{T_{ij}\left(U_{ij}-L_{ij}\right)},\quad \frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial U_{ij}}=\frac{1}{1+e^{-x_{ij}/T_{ij}}}$$

$$\frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial L_{ij}}=1-\frac{1}{1+e^{-x_{ij}/T_{ij}}},..\ \frac{\partial f\left[x_{ij},U_{ij},L_{ij},T_{ij}\right]}{\partial T_{ij}}=\frac{x_{ij}\left(O_{ij}-U_{ij}\right)\left(O_{ij}-L_{ij}\right)}{T_{ij}^{2}\left(U_{ij}-L_{ij}\right)}$$

as opposed to the derivative

$$\frac{\partial f\left[x_{ij}\right]}{\partial x_{ij}} = \frac{e^{-x_{ij}}}{\left(1+e^{-x_{ij}}\right)^2}$$

in the BP algorithm. In the above equations, $O_{i,j}$ represents the generic output of a neuron at the hidden layer(s), and at the output layer respectively.

The previous example illustrated successful applications of neural identifiers. Next, the problem of adaptive control via NNs is considered. The problem of controlling a generic dynamic system can be divided into a regulation or a tracking. In a regulation problem the goal is to achieve, through an appropriate feedback control law, an equilibrium condition around a certain set point. In the tracking problem the objective is instead to achieve, through the feedback control law, an accurate following of a reference model response. The control part of this example is related to the tracking problem for an unknown non-linear SISO system. It is known that most of adaptive control theory is based on the Model Reference Adaptive Control (MRAC) relative to linear time-invariant systems. Particularly, Reference 14 shows two different approaches, that is the direct and the indirect approaches. It should be emphasized that the closed-loop dynamics with the adaptive control scheme, both in the direct and indirect form, will be nonlinear even if the open-loop system is linear and time-invariant. In a direct adaptive control scheme the parameters of the control scheme are adjusted on-line to satisfy some control specifications. In indirect adaptive control scheme instead the parameter identification scheme is first implemented providing estimates of the actual system parameters with the parameters of the controller calculated assuming these estimates of the system parameters.

There is a well known problem in using a neural network approach for a direct adaptive control of a non-linear system. In fact, the neural network controller output error, which is used to update the weights and the thresholds of the neural controller, is not directly available. The only available information is the tracking error between the desired and the actual response, which is only a function of the error at the neural controller output. Several methods have been proposed in the literature to overcome this problem and tested on non-linear SISO systems. Reference 21 suggests a method where the system dynamics are treated as not modifiable layer of the neural controller and the tracking error is back-propagated through the dynamics. In [5] another approach is suggested, where the tracking error is transformed into the neural controller error using the dynamics inverse transfer matrix. Reference [12] presents a third method where a feedback gain is introduced to generate a transformed error signal used to update the architecture of the neural controller. Finally in [26] a scheme is introduced, where the architecture of the neural controller is updated using the tracking error with the addition of an extra single layer gain between the neural controller and the system.

This section, however, describes results related to indirect adaptive control with a neural approach. In the absence of well formulated stability and robustness criteria a natural approach for nonlinear systems can be given by trying to extend some results relative to linear adaptive control theory. For example, in linear adaptive control theory it was concluded that adaptive control laws for a stable closed-loop system can be found if some "a priori" knowledge of the system was available. Similarly, the probability of success of the adaptive control of a nonlinear SISO system can be assumed to be a strong function of the amount of "a priori" information on the unknown system.

A safer and more conservative set-up of the problem is to apply adaptive control, in order to achieve tracking of a desired system response using an indirect approach. That is only after the non-linear system has been identified within a desirable level of accuracy. It should be underlined, however, that even this conservative approach does not guarantee closed-loop stability and acceptable tracking performance, even in the case of a BIBO stable system, since the adaptive control law can lead to an unstable system during the transient. It can be speculated that fast on-line learning capabilities of the neural controllers, which are direct function of the selected learning algorithm, are critical for this purpose. On the other side, closed-loop stability, and possibly good tracking performance are only a remote possibility if adaptive control using neural identifiers and neural controllers is attempted with a direct approach, that is without any "a priori" identification phase. In the following, the tracking capabilities and the robustness of a NN-based indirect adaptive control scheme are evaluated.

Consider the same unknown SISO nonlinear system described before, and representing a Model III system. This time the objective is to find an adaptive control law $u(k)$ such that the actual system response $y(k)$ tracks a desired model response $y_m(k)$, with the dynamics for the reference model given by:

$$y_m(k+1) = 0.6y_m(k) + r(k)$$

recall that previously we had:

$$\hat{y}(k+1) = N_f[y(k)] + N_g[u(k)] = \hat{f}[y(k)] + \hat{g}[u(k)]$$

Following the training phase we had that the estimates of $y(k)$ were converging to the actual $y(k)$ response. For the actual system to track the desired model response we would have

$$\hat{f}[y(k)] + \hat{g}[u(k)] = 0.6y_m(k) + r(k)$$

isolating $\hat{g}[u(k)]$ yields:

$$u(k) = \hat{g}^{-1}\left\{0.6y_m(k) + r(k) - \hat{f}[y(k)]\right\} = \hat{g}^{-1}[z(k)]$$

The tracking problem is solved by introducing a neural adaptive controller $N_c$ for the online determination of $u(k)$. In particularly, two neural controllers, trained using EBP and BP respectively, are introduced. Both of them will have architecture $N_{1,20,10,1}$ with input $z(k)$ with a learning rate of 0.01. The closed-loop system will therefore include two neural identifiers ($N_f$,$N_g$); and one neural controller ($N_c$).

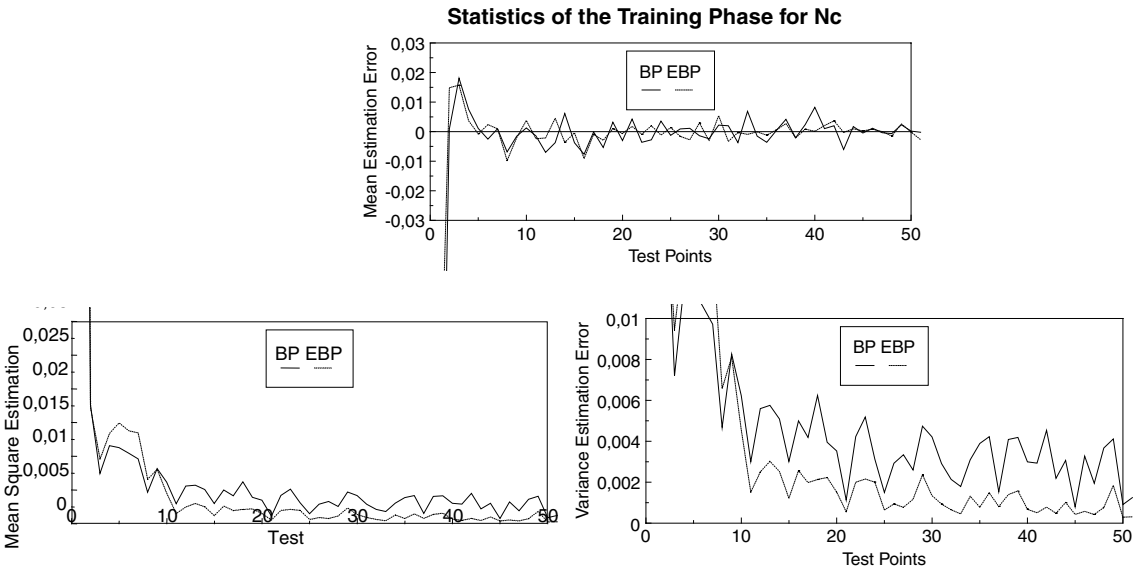


Figure 4. Statistics of the Training Phase for "g-inverse"

Again, there is an initial training phase and a testing phase. During the training phase two similar sets (one using BP and the other using EBP) of 3 neural architectures ($N_f$,$N_g$,$N_c$) were trained for 100,000 steps using the same random input $u(k)$ between [-2,2] used previously. The training performance were evaluated every 2,000 steps for a total of 50 test points. The statistical results for the identification of *'f'* and *'g'* were very similar to the ones obtained previously, while the results for the identification of the inverse function of *'g'* are shown in Figure 4. The tracking capabilities of the closed-loop system are then evaluated with a 5000 steps simulation. The results are summarized in Figure 5; in particular Figure 5 shows 2 different 100 steps simulation starting respectively at instants 1, and 4,901. The superior performance of the EBP algorithm are once again confirmed by the statistical data on the tracking error shown in Table II. It can be seen that the EBP neural estimate is practically unbiased and has a variance approximately 4.25 times smaller than the correspondent BP variance.

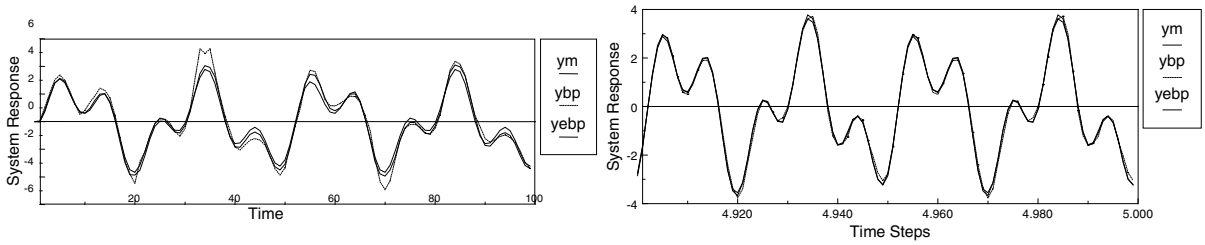


Figure 5. Testing Phase: Simulation

Table II. Statistics of the Tracking Error: Testing Phase

| | BP Algorithm | | |
|---|---|---|---|
| Time | 1-1,000 | 2,501-2,600 | 4,901-5000 |
| Mte | -0.0237 | .0004 | -.0003 |
| MSte | .2743 | .0139 | .0148 |
| Vte | .2765 | .014 | .0149 |
| | EBP Algorithm | | |
| Time | 1-1,000 | 2,501-2,600 | 4,901-5000 |
| Mte | .0707 | .0005 | .0002 |
| MSte | .0720 | .0042 | .0035 |
| Vte | .0677 | .0042 | .0035 |

The next step of this study is to evaluate the robustness of the neural-based tracking to a time-varying system. For this purpose the original nonlinear SISO system is simulated to change instantaneously, starting at the 5,000-th step, to 3 different arbitrarily selected configurations before returning to the original configuration. At each configuration a 5,000 steps simulation is performed using both the EBP and the BP algorithms. The different configurations are:

For k=5,000-9,999, Config. 1

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+\left[1.5u(k)\right]^3+2$$

For k=10,000-14,999, Config. 2

$$y(k+1)=\frac{2y(k)+2}{0.5+y^2(k)}+\left[1+u(k)\right]^3-1$$

For k=15,000-19,999, Config. 3

$$y(k+1)=\frac{2y(k)+2}{0.5+y^2(k)}+\left[1+u(k)\right]^3-1$$

For k=20,000-24,999, Config. 4

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+\left[u(k)\right]^3$$

Several 100 steps simulations time histories of the reference model along with the closed-loop system response using both EBP and BP algorithms are shown in the next figures for different points of the 5,000 step simulation. For the tracking problem, the statistics are expressed in terms of the tracking error, defined as:

$$M_{TE}=\sum_{i=1}^{100}\frac{\left[y(i)-y_m(i)\right]}{100}=\sum_{i=1}^{100}\frac{e(i)}{100},\ MS_{TE}=\sum_{i=1}^{100}\frac{\left[y(i)-y_m(i)\right]^2}{100}=\sum_{i=1}^{100}\frac{e(i)^2}{100},\ V_{TE}=\sum_{i=1}^{100}\frac{\left[e(i)-M_{TE}(i)\right]^2}{100}$$

From the plots and the statistical results, it can be seen that the tracking performance associated with the EBP algorithm are somewhat worse than the correspondent BP performance only during the transient phase of the transition of the system between different configurations. Once this transient phase is past, the closed-loop system with EBP trained neural identifiers and controller clearly outperform the BP version.

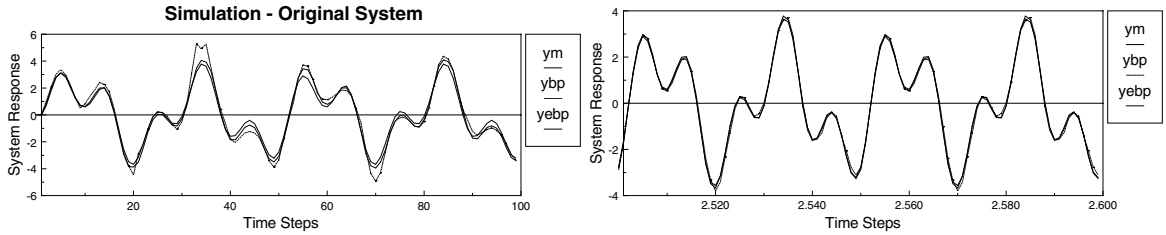


Figure 6. Simulation Comparison Original System

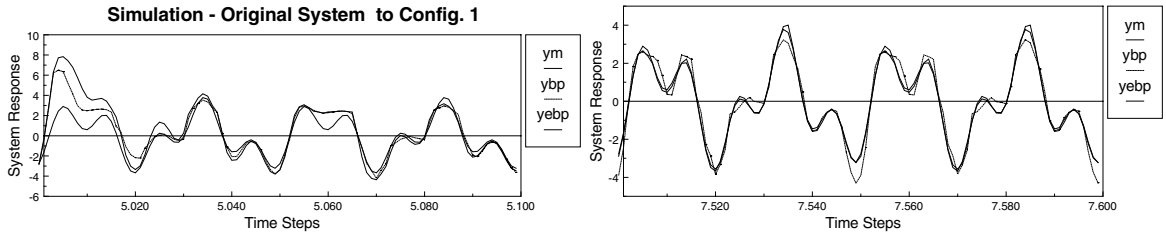


Figure 7. Simulation Comparison to Config. 1

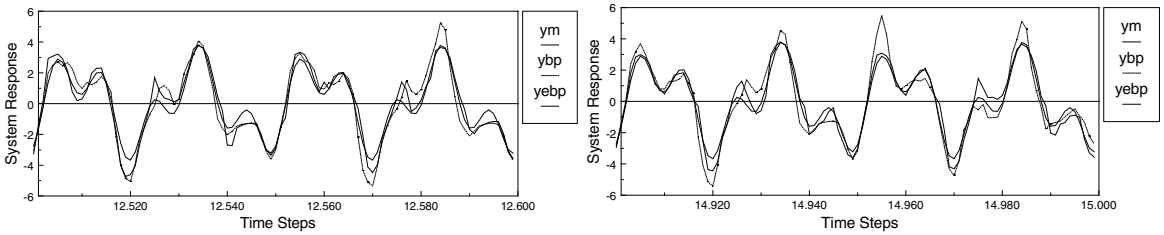


Figure 8. Simulation Comparison to Config. 2

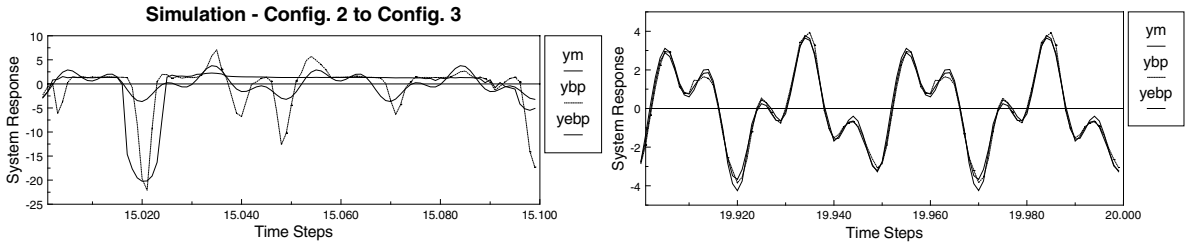


Figure 9. Simulation Comparison to Config. 3

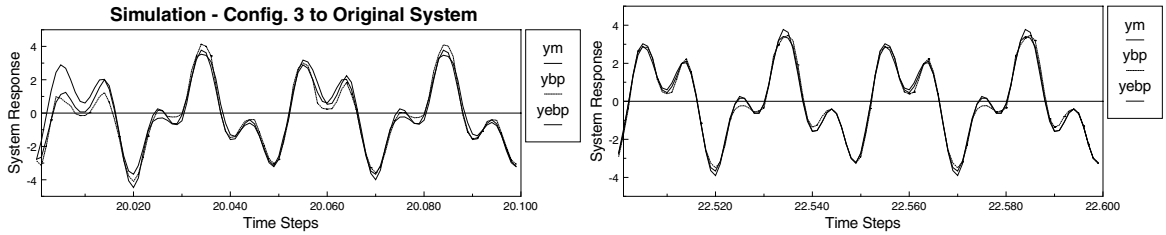


Figure 10. Simulation Comparison to Original System

## Failure Detection using Neural Networks

Failure detection (FD) is a widely investigated problem in current flight control research, due to the ever-increasing complexity, and interconnectivity of the system at end. Generally speaking, FD implies some sort of continuous monitoring of the measurable outputs of the system. Under nominal conditions, these variable

tend to follow rather well established patterns, within the uncertainty due to process disturbances, such as atmospheric turbulence, sensor noise, and actuator biases. A failure to any component of the flight control system (FCS) will produce deviations from nominal and somewhat predictable trajectories. Spotting a deviation between nominal and failed trajectory is usually the basis of any FD technique. The knowledge of the system's behavior by the flight control system at any operating point needs then to be available at a very heavy computational and storage costs, limiting the prediction to linear time invariant, low order, and white noise-based models of predicted trajectory behavior.

Neural network architectures have been primary candidates to maintaining the same level of, or improving FD capability without increasing computational burden. The mapping property of a NN architecture can be extremely attractive when the input-output data are related to completely or partially unknown dynamics. Several factors must be taken into account however, when considering the implementation of NN estimators. A key issue in the design of a NN based state estimator is the selection of online learning vs. offline learning. Of course in an offline situation all training has been performed beforehand, and the network has a frozen architecture. Due to the variability of the dynamics and failure conditions, however, the potential of on-line architectures was studied in the present work because of its inherent added flexibility. In the following, we refer to an on-line structure of feed-forward type with interlayer connections schematically shown in Figure 1 as implemented by a three-layer NN. More details on the architecture can be found in [32].

*General Sensor Failure Detection*

The overall sensor failure detection, identification, and accommodation (SFDIA) problem is addressed in the context of aircraft dynamics, by using multiple architectures consisting of a centralized structure (MNN), and a set of 'n' decentralized structures (DNN), where 'n' is the number of non-redundant sensors. Figure 11 shows a diagram of the layout of the main neural network. A similar one, but with a single output, can be obtained for the n DNNs which, after sensor failure, will provide accommodation. Figure 11 shows the flow of the SFDIA process in a three-sensor case, with a simulated failure for sensor #2. A more detailed description of the process can be found in the references (32, 35).

The main network MNN receives the input from control commands and the vector of measurements $\underline{y}$, providing the estimate $\underline{y}^E$ of the same measurements as output. The EBPA updates MNN based on the error, and the learning process is monitored by a quadratic error parameter

$$E_{MNN} = \sum_{i=1}^{outputs} \left( \underline{y}^E - \underline{y} \right)^2$$

Each $i^{th}$ DNN receives the input form the control and *n-1* sensors, where the missing signal is relative to the corresponding measurement. The training procedure is similar to that of MNN, until failure is detected. Upon failure of the $i^{th}$ sensor, detection can be accomplished by the value of $E_{MNN}$ exceeding a given threshold.

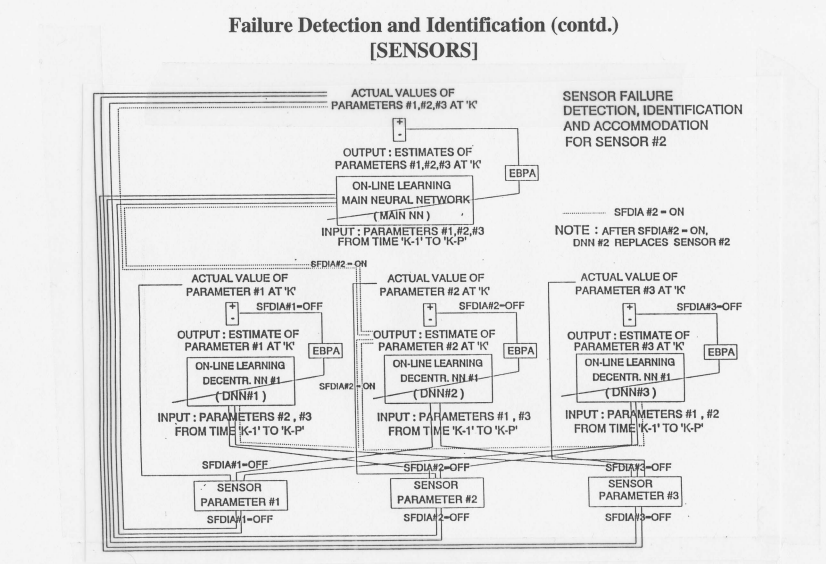


Figure 11. SFDIA Diagram, Example of failed #2 Sensor

When this occurs, identification can be achieved by monitoring the absolute value of the estimation error of the $i^{th}$ DNN

$$E_{DNN}^{ID} = \left| E_j \right|$$

Identification is again assumed if the error in the above equation is greater than a specified level. The procedure outlined above consists of two steps, thus reducing the chance of false alarm. Failure accommodation is then performed by replacing the value of the data coming from the failed sensor, with the estimate of the corresponding DNN, which stops the learning process and "freezes" its architecture at this point.

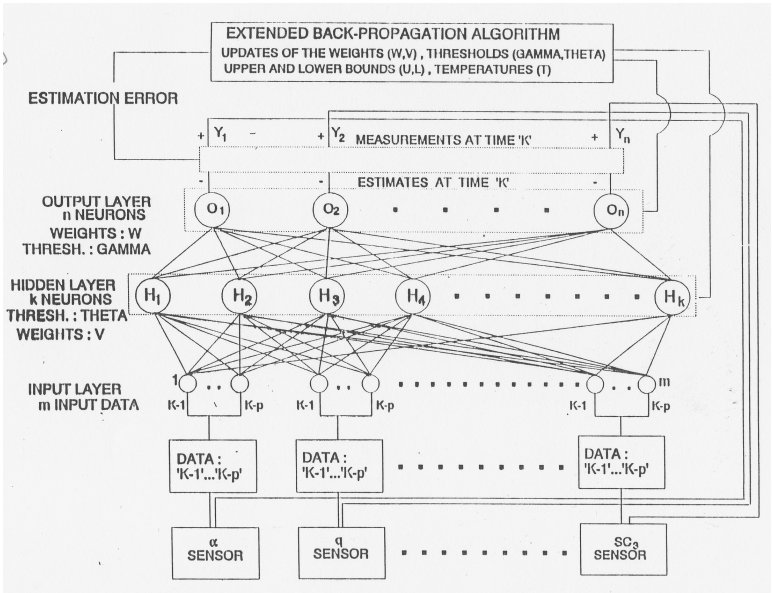


Figure 12. Sensor Failure Accomodation

Figure 12 with a single output describes therefore the accommodation procedure as well, whereas the entire SFDIA is described in Figure 11, noting that DNNs update is performed at each step, at the frequency of the sensors. References 34, and 35 present several applications simulating soft and hard failures in the dynamic behavior of a full-envelope nonlinear aircraft model representative of a high performance twin engine flight vehicle. The system considered here has 11 sensors and 5 actuators. As an example, we consider the case of a soft failure (the sensor does not fail completely) representative for instance of added bias in the instrument

readings. The sensor considered here is the roll-rate gyro, which is one of the primary measurements in the lateral-directional motion of the aircraft. Figure 13 shows the time history of the error in the MNN with a detection spike at 1700 seconds, while the measured variable, roll rate is plotted in Figure 14. From that figure, the time history of the roll rate DNN follows the actual roll rate in a satisfactory manner.

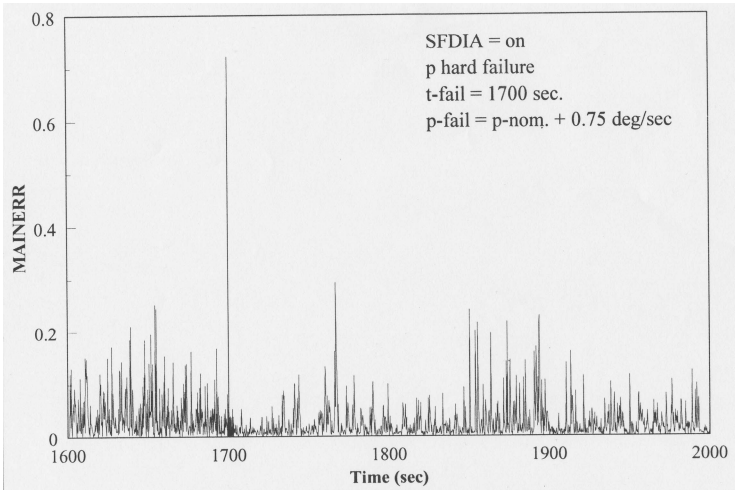


Figure 13. Roll Rate Gyro Bias Failure, NN Est. Error, SFDIA on

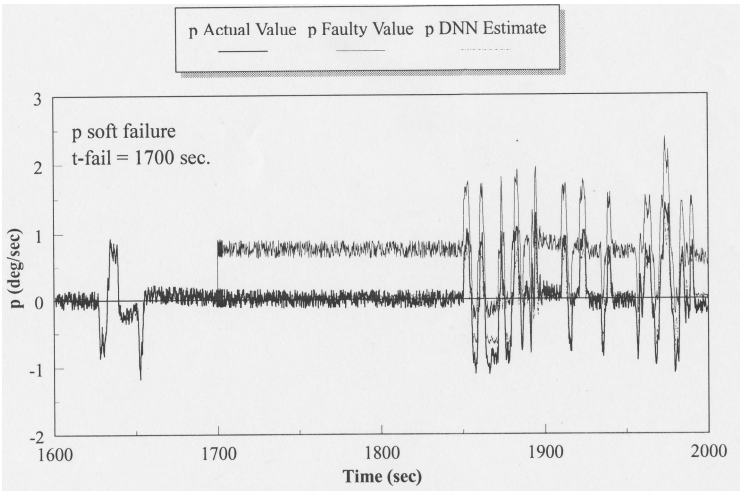


Figure 14. Roll Rate Gyro Failure, Time Histories

*General Actuator Failure Detection*

During the operation of a flight vehicle, actuator failures can occur for reasons internal to the system, as well as for external causes such as battle damage for military aircraft. Dealing properly with actuator failure is usually critical in that reduced controllability and even loss of controllability may happen with heavy financial losses as well life-threatening situations. The same dynamic system described in the previous section is considered here, which is modeled by a full nonlinear simulation code, and a FCS consisting of five aerodynamic control surfaces providing control moments on the three axes (pitch, roll, and yaw). The simulation is related to a typical high maneuverability task with simultaneous commands on the actuators, plus the appropriate thrust setting. Several actuator failures have been studied involving a fault in the hinge moment capability, and percent loss of control effectiveness. These failures require extensive analysis of the modified aerodynamic effects on the aircraft, in order to have a realistic simulation capability and were part of a concerted research effort, as reported in Reference 35.

The measurement data are all fed into the neural network, which is training on-line, and the process is monitored by evaluating the difference between the estimated and actual angular velocity in terms of body axes components at each instant *k* as

$$\Omega_{err}(k) = \frac{1}{2}\left[\left(p(k)-p^{est}(k)\right)^2+\left(q(k)-q^{est}(k)\right)^2+\left(r(k)-r^{est}(k)\right)^2\right]$$

It must be noted that in this case, unlike SFDIA, the network must be able to separate sudden variations in the actuator time histories due to pilot maneuvering commands, from failures, therefore training plays a much more critical role here. A parametric analysis was carried out to determine the most appropriate failure detection and identification scheme keeping in mind a compromise between NN architecture complexity and on-line computational requirements. The results led to a structure consisting of 9 inputs 2 patterns, 18 input data, 1 hidden layer, 10 P.E. in the hidden layer, and 3 P.E. in the output layer.

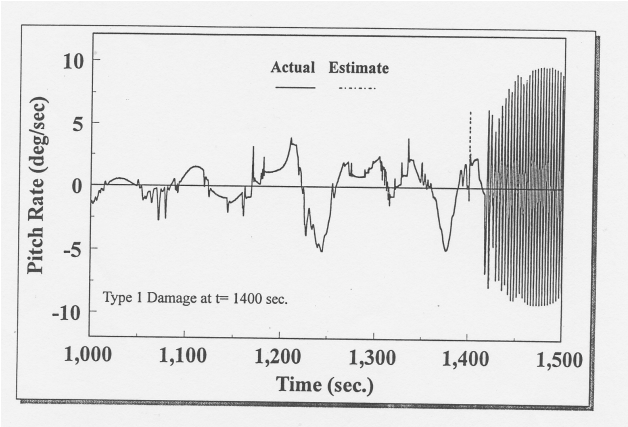


Figure 15. True and Estimated Pitch Rate after Failure

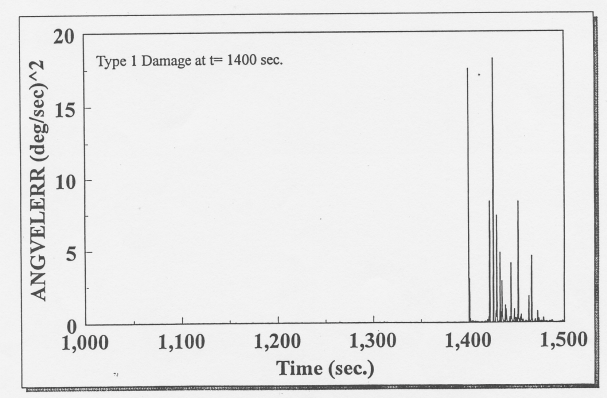


Figure 16. Actuator Failure Detection using Angular Velocity Error

Figure 15 gives an example of pitch rate ($q$) time history following a damage of the left stabilator of the aircraft at t=1400 sec (control surface stuck at 20 degrees, and 50% loss of control effectiveness). The change in behavior of $q$, which is coupled with similar changes in the other components of angular velocity, induces a variation in $\Omega_{err}(k)$, interpreted as failure detection as shown in Figure 16. Note that a different parameter could have been chosen in the failure detection scheme. Choice of $\Omega_{err}(k)$ was preferred since it contains variables directly affected by the aircraft dynamics, and dimensionally consistent. Following the detection of failure, sensor data must be interpreted in order to identify the damage location. The identification procedure

relies on on-line evaluation of the cross-correlation function between key variables, which is then stored in temporary memory on the FCS computer. In the above example of a damaged stabilator, since both pitch and roll rate are involved, the cross-correlation function used is given by

$$\sum \left| R_{pq}^{coef}(k-1) \right| - \sum \left| R_{pq}^{coef}(k-2) \right| > \varepsilon_{FI}^{Thres}$$

and similarly for the other actuators. The behavior of the cross correlation time history is shown in Figure 17 taken with a time window of 7 instances. Although the presence of false alarms is not avoided, especially at the detection level, the use of above equation has proven very effective in terms of false alarm rate of the entire failure detection and identification process.

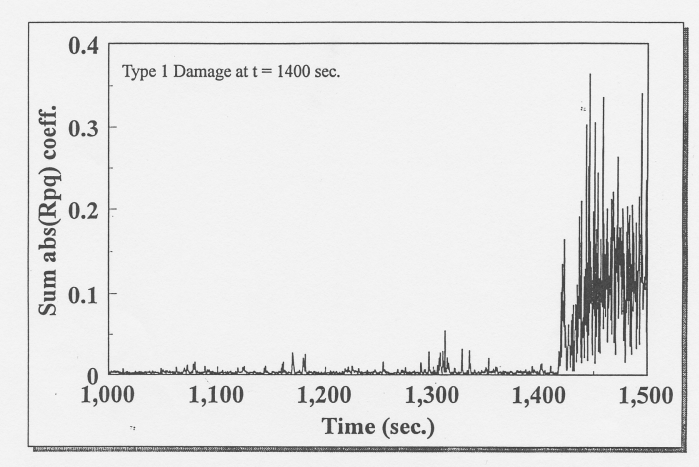


Figure 17. Pitch Rate and Roll Rate Cross Correlation as Failure Identification Parameter

*Comparison with Standard Methods*

One of the techniques traditionally used is the failure detection problem is based on the application of a Kalman filter both in its linear (KF) and nonlinear (EKF) versions. Valuable contribution of Kalman filtering relies heavily on the particular structure of the dynamic system, that is its linearity or linearization properties. In the following, a simple comparison between the neural network approach presented and the application of KF is presented. The dynamic system considered is representative of a high altitude unmanned aircraft used for scientific missions, sensor failure detection and identification is analyzed, based on a linearized model of the system itself, and relative to angular velocity rate gyros. The structure of the KF-based scheme is taken at first as specular of that relative to the neural network SFDIA previously described This implies a main Kalman filter block (MKF), plus a number *n* of decentralized blocks (DKFs) corresponding to the number of on board sensors (three sensors in this application). Several types of failures were considered ranging from instantaneous biases of different amplitude, to biases entered with ramp transients. Different system's configurations were also studied, from nominal model and system dynamics and noise, to discrepancies in the dynamics and noise statistics, and detailed results can be found in references 37, and 38. The example in figure 19 shows the failure of the Kalman filter scheme to recognize a 2.4 deg/sec bias entering as a ramp as supposed to the NN scheme, when system and filter models have discrepancies in the dynamics and noise statistics.

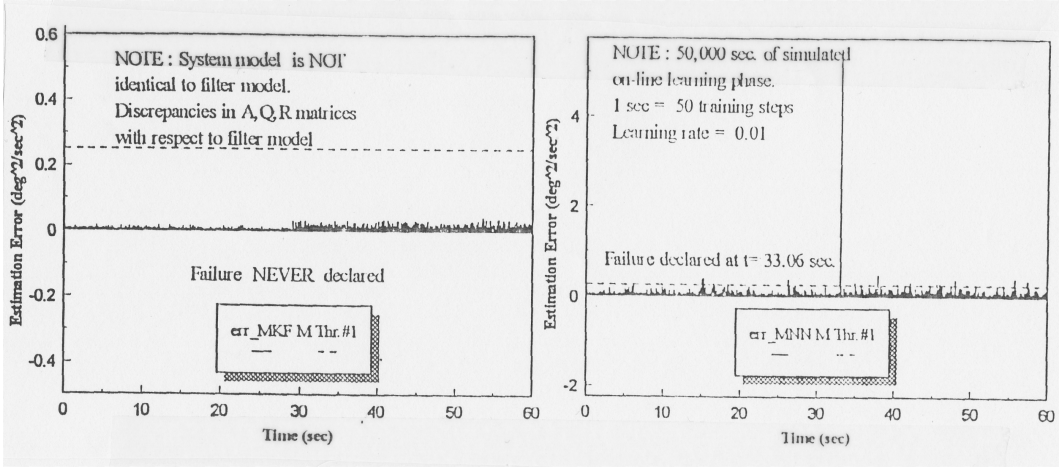


Figure 18. Estimation Error Comparison between KF and NN, with Model Discrepancy

**Sensor Validation using Unmanned Air Vehicle Data**

This section describes some results of SFDIA research besed on experimental data relative to an unmmanned research aircraft.

*Description Of The B777 Model And Its Electronic Payload*

The aircraft model shown in Figure 19 is a 1/24$^{th}$ semi-scale B777 research designed, built, assembled, instrumented, and flown at WVU. Scaled research models do not exhibit acceptable flying qualities due to the negative effect of the payload on the dynamic characteristic. Therefore, specific changes were made to the geometric and aerodynamic characteristics of the "perfectly scaled" model to achieve desirable handling qualities. In particular, the design aimed to achieve acceptable values for two specific parameters, that is the Thrust/Weight (T/W) and the Weight/Wing Surface (W/S) ratios. The aircraft was essentially designed around its electronic payload with a simple "2 component: tail/fuselage + wing" architecture. The wing, featuring ailerons, inboard/outboard flaps and housing the fuel tanks, was manufactured with fiberglass, carbon fiber, foam, and lightweight plywood.

Figure 19 - B777 Model in Flight and during Approach

The fuselage was manufactured using carbon fiber as a whole piece along with the horizontal tail and the vertical tail. Additional servos on the fuselage include a nose wheel and a brake servo. The B777 aircraft propulsion system features two ducted fan engines with a nine-rotor blade system with an engine shroud/rotor hub assembly. The units are housed in scaled engine nacelles to add realism to the model. The main characteristics are summarized in the following Table.

| | |
|---|---|
| Fuselage length | 8.75 ft |
| Wing span | 8.92 ft |
| Wing surface | 11.3 $ft^2$ |
| Weight (with R/C components, 48 oz. Fuel, and payload) | 46 lb. |
| Maximum thrust (approx.) | 24.0 lb. |
| Weight (with R/C components, 48 oz. fuel) | 30.2 lb. |

Table III – Research Model Characteristics

The onboard computer package, featuring a CPU board, a passive backplane, a 16 MB RAM Disk card, a data acquisition card and a chassis frame, is the core of the aircraft electronic payload since it executes the data acquisition software to acquire the data from all the sensors. The 16 MB RAM disk stores the on-board software and the recorded flight data, which, upon landing, are retrieved through laptop computer via parallel port. The aircraft transmitter/receiver unit features a 10-channel programmable/menu driven system with the capabilities of customizing aircraft controls with a built-in microprocessor. This ground unit allows the pilot to command the conventional aircraft controls with the addition of mixing control surfaces and programming maneuver functions. In terms of the sensor capabilities, the electronic payload features a unit with 3 gyros and 3 accelerometers, an air-data probe with pressure sensors for altitude and airspeed measurements, an aerodynamic vane with potentiometers for the aerodynamic angles, and potentiometers at the hinges of each control surface for the measurement of the surface deflections. Additional components of the payload include batteries and power converters. A total of 33 flights have been performed with 16 flight tests specifically for SFDIA purposes.

*Learning-Based SFDIA*

Analytical redundancy based methods have been widely discussed in the past. Specific applications have also been developed in the field of flight control systems. In these methods an analytical model of the system is used inside within "observers" to provide estimates of measured variables. This redundancy is then used both to detect a fault and to provide fault tolerance capability to the system.

The most relevant signals that characterize the longitudinal dynamics of an aircraft are:
Angle of attack $\alpha$ [rad];
Pitch rate $q$ [rad/sec];
Normal acceleration $a_z$ [g];
Elevator deflection $\delta_e$ [rad].
Velocity relative to the air V [m/s]
Altitude H [m]
The analytical redundancy between these signals is ensured by the equations of motion of the aircraft and in particular by the "normal force equation" below expressed along the body fixed reference frame:

$$ma_z = \rho(H)V^2SC_z(\alpha,\delta_e,V,a_z/V^2,q/V)/2$$

where $m$ is the aircraft mass, $a_z$ is the normal acceleration measured at the center of gravity, $S$ is the reference surface, $\rho$ is the air density, and $Cz$ is the normal force non-dimensional coefficient (which is a function of $\alpha$, $q/V$ and $\delta_e$). In this effort it was assumed - without any loss of generality - that a failure can occur only on the sensors measuring $\alpha$, $q$, and $a_z$. The analytical redundancy between the variables involved in the above has

been exploited to build up the input-out models required in the SFDIA scheme. In particular, the following estimation models where employed:

$$\hat{\alpha}(k+1) = f_{\alpha}\left(V(k), H(k), a_z(k)/V^2(k), q(k)/V(k), \delta_e(k)\right)$$
$$\hat{q}(k+1) = f_{q}\left(H(k), V(k), a_z(k)/V^2(k), \alpha(k), \delta_e(k)\right)$$
$$\hat{a}_z(k+1) = f_{a_z}\left(H(k), V(k), q(k)/V(k), \alpha(k), \delta_e(k)\right)$$

The signals $a_z/V^2$ and $q/V$ have been used in the above relationships instead of $a_z$ and $q$ because they allow for the non-linearities of the functions to be smoother and, therefore, easier to be identified. In general, since the non-linearities in aircraft dynamics are relevant, the use of non-linear approximators is required.

The purpose of residual generation is to check if the actual measurements fit the process model. This can be accomplished by analysing the trend of the following residual signals $r_i(k)$ :

$$r_{\alpha}(k) = \alpha(k) - \hat{\alpha}(k) + n_{\alpha}(k) + F_{\alpha}(k - k_f)$$
$$r_{q}(k) = q(k) - \hat{q}(k) + n_{q}(k) + F_{q}(k - k_f)$$
$$r_{a_z}(k) = a_z(k) - \hat{a}_z(k) + n_{a_z}(k) + F_{a_z}(k - k_f)$$

where $n_i(k)$ is the measurement noise and $F_i(k\text{-}k_f)$ is the additive fault modeling function. Typically this function is different from zero after the occurrence of the fault at the instant $k=k_f$. The residual set above has not been designed to exhibit predefined sensitivities to different faults. In other words a single measured quantity should not have impact on a specific residual, based on the structured residuals generation approach. Therefore the models expressed in the two sets of equations can be used as "virtual sensors" or as "residual generators" respectively.

Typically the failure modelling function $F(k\text{-}k_f)$ is described by step and ramp functions representing abrupt and incipient faults (bias or drift) respectively. Within this effort a linear ramp-like function is assumed. Thus, the failure is modelled as follows:

$$F(k-k_f) = \begin{cases} 0 & k < k_f \\ A \cdot (k-k_f)/T_R & k_f \le k < k_f + T_R \\ A & k \ge k_f + T_R \end{cases}$$

where $T_R$ is the duration of the ramp and $A$ is the final value. By varying $T_R$ it is possible to model either hard or soft failures.

The filtered residual vector is processed by an isolation and accommodation logic in order to infer information about the health status of sensors. When the squared filtered residual exceeds a predefined threshold $S_{10}$, the state of the corresponding sensor is declared suspect (fault *detection*) and a procedure is invoked to decide on the health status of this sensor. At the same time the learning of the NN is stopped to avoid that the NN learns from the "failure-contaminated" measurement. If the filtered residual exceeds another threshold $S_{20}$ the state of the sensor is then declared faulty (fault *declaration*), a failure procedure is enabled, and an accommodated variable $y_a(k)$ is provided as output. Thus, the accommodation procedure substitutes the faulty measurement with the estimation given by the NN. As for any SFDIA approach, the following capabilities are critical:

Failure *detectability* and false alarm rate (the sooner the fault is detected and the least the number of false alarm it is, the better is the SFDI system).
*Estimation error* (The least is the estimation error, the better is the quality of the accommodation).

The used fault isolation and accommodation scheme is only a step toward the development of a realistic and reliable online scheme, on the other hand, the main purpose of this paper was to investigate, using real flight data, the suitability of different online neural approximators for fault detection purposes rather than focusing on failure identification and accommodation issues.

*Neural Networks for On-line Approximation*

The two main classes of neural architectures that have emerged in the literature are the Multi Layer Perceptron (MLP) NN and the Radial Basis Function (RBF) Networks NN.

Multi-Layer Perceptron with Extended Back-Propagation (MLP-EBP).

It is a 3-layer architecture NN, whose detailed description has been already done in the paper, with the following sigmoid activation function at each of the neurons:

$$f(net,U,L,T) = \frac{U-L}{1+e^{-net/T}} + L$$

where *U*,*L*, and *T* are the upper bound, the lower bound and the slope at the origin of the activation function respectively. The EBPA updates not only the matrices of the weights between input/hidden layers and hidden/output layers - *W(k)* and *V(k)* respectively - but also the parameters *U,L,T* at the neurons at the hidden and output layers.

"Conventional" Radial Basis Function (RBF)

In the "conventional" RBF NN the estimations $y_s \in \Re^m$ are expressed as a linear combination of *M* Gaussian Basis functions:

$$y_s(x) = We^{-\frac{(x-\mu)^T(x-\mu)}{2\sigma^2}}$$

where $x \in \Re^n$ is the input vector, the parameters $\mu_j$ and $\sigma_j$ are the basis center and width respectively. In the conventional implementation the hidden layer neurons are *a priori* statically allocated on a uniform grid that covers the whole input space and only the weight $w_{ij}$ are updated. This approach requires an *exponentially increasing* number of basis functions versus the dimension of the input space.

Fully Tuned Extended Minimal Resource Allocation Network RBF (EMRAN-RBF).

To avoid the dimensionality problems associated with the conventional RBF, a sequential learning technique for RBF NNs has been introduced in the literature. The resulting architecture was named the Resource Allocating Network (RAN) and has shown to be suitable for online modeling of non-stationary processes with only an incremental growth in model complexity. The RAN learning algorithm proceeds as follows: At each sampling instant, new neurons are added if all the following 3 criteria are met:

*Current estimation error criteria*: error must be bigger than a threshold:

$$e(k) = y(k) - \hat{y}(k) \geq E_1$$

*Novelty criteria:* the nearest center distance must be bigger than a threshold:

$$\inf_{j=1}^{M} \left\| x(k) - \mu_j(k) \right\| \geq E_2$$

*Windowed mean error criteria:* windowed mean error must be bigger than a threshold:

$$1/T \sum_{i=0}^{T} [y(k-T+i) - \hat{y}(k-T+i)] \geq E_3$$

If one (or more) of the above criteria are not met, the existing network parameters (the centers $\mu_j$, the weights $w_{ij}$ and the variances $\sigma_j$) are adjusted using the on-line learning algorithm. To avoid an excessive increase of

the Network size a *pruning strategy* can also be applied leading to the so-called Minimal RAN (MRAN). Furthermore, the adaptation algorithm is called Extended MRAN (EMRAN) when the parameters are updated following a "winner takes it all" strategy. More precisely, using the EMRAN algorithm only the parameters of the most activated neurons are updated, while all the others are unchanged. This strategy implies a significant reduction of the number of parameters to be updated online with just small performance degradation with respect to the MRAN.

*Application to the WVU B777 Flight Data*

Four flight data windows of approximately 200 sec each from 4 different flights of the model were used for the SFDIA experiment. The first two sets of flight data were used for off line training purposes while the last two were used for validation purposes. Figure 20 shows a typical time history of *q(k)* and its estimation, during the occurrence of a simulated failure on the pitch gyro at time $t_f$ =150 s using the EMRAN algorithm. To highlight the effects of the failure a "large bias" failure type was selected (A=10 deg/sec $T_R$=1 sec).

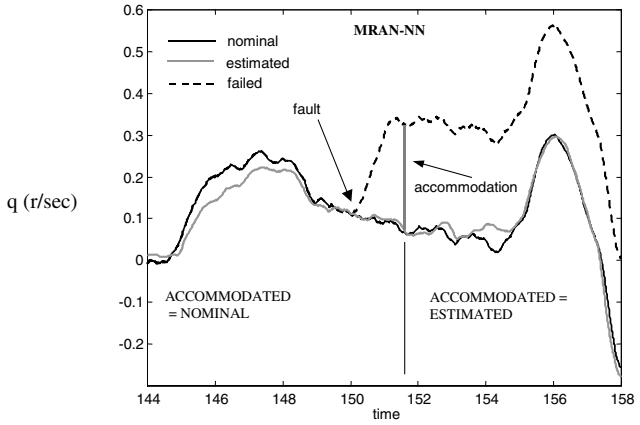


Figure 20 - Failure on the q gyro at $t_f$ = 150 sec

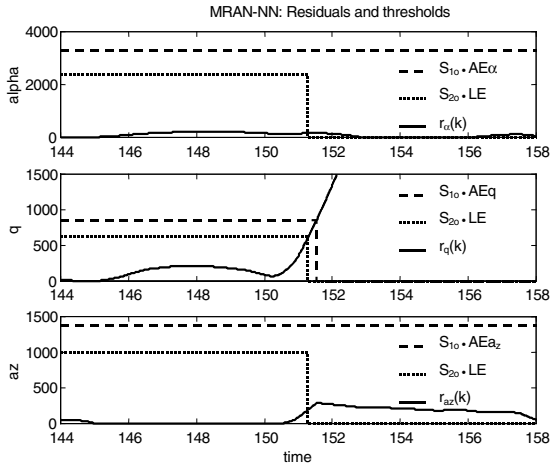


Figure 21 - Residuals associated with a failure on the q gyro at $t_f$ = 150

Figure 21 shows the time histories of the residual signals. The three residuals are generated subtracting the three EMRAN-generated estimations, respectively to q, $\alpha$, and $a_z$. After the failure the residuals increase. However, due to the different sensitivities to a fault on the q gyro, only *r_q(k)* exceeds the thresholds. In particular at t=151.2 s the threshold S1o is exceeded; therefore the learning of the three NNs is preventively halted. At t=151.5 s the failure on the sensor *q(k)* is "officially" declared since *r_q(k)* also exceeds the threshold S2o. From this moment on, the on-line neural estimate of *q(k)* is used in lieu of the measurement from the faulty sensor. The failure identification was possible since the residuals *r_az(k)* and *r_α(k)* remain well below their thresholds.

*Neural Networks Comparison*

The capabilities of detecting small amplitude and slowly drifting bias failures are strictly related to three aspects:

level of noise in the measured signal;
level of accuracy of the estimator;
level of the on-line learning rate.

While the presence of measurement noise is independent of the implemented algorithm, the other two aspects are critically dependent on the performance of the selected estimator. In particular, the selection of the learning rate is reached as the result of a trade-off since it should be high enough to allow the learning of a new aircraft operating condition while, at the same time, small enough to avoid the learning of a faulty measurement from slow drifting fault. On this basis, the performances of the two different NN architectures (EMRAN and MLP-EBP) were compared by varying the level of the learning rate on the data of one validation flight, where at the time $t=t_f=90$ sec a step of 5 deg/sec (7.4% of the maximum value of the pitch rate) was artificially injected on the measured q signal to simulate the effect of a sensor failure.

The comparison of the performance of the two schemes is performed through the evaluation of different SFDIA indicators: TLE and TAE. TLE is the time in which a sensor is declared suspect while TAE is the time at which it is finally declared faulty. To evaluate the performance of the detection/identification system, the detectability ratio (DR) between the main peak of the filtered residual signal during the failure transient ($90 < t < 100$) and the peak of the filtered residual before failure ($0 < t < 90$) is considered. This ratio quantifies the detectability provided by the scheme.

Table IV – EMRAN-RBF; $\eta o=0.0001$

| ***L.R.(*** ŋ | 0 | ηo | 2ησ | 4ηo |
|---|---|---|---|---|
| TLE | 93 | 93.39 | 93.71 | 94.71 |
| TAE | 93.89 | 94.25 | 94.97 | 95.95 |
| ***%LE*** | 0 | 0 | 0 | 25.14737 |
| ***%AE*** | 0 | 0 | 0 | 25.61053 |
| ***DR*** | 11.63651 | 11.49651 | 12.35879 | 12.19832 |
| ***MEE*** | -1.73646 | -2.04047 | -2.39899 | -3.95824 |
| ***STDEE*** | 2.452785 | 2.407548 | 2.369341 | 2.202287 |
| ***POWEE*** | 9.031454 | 9.959815 | 11.36895 | 20.51775 |

Table V – MLP-EBPA; $\eta o=0.00006$

| ***L.R. (*** ŋ | 0 | ηo | 2ησ | 4ηo |
|---|---|---|---|---|
| TLE | 94.74 | 94.22 | 94.32 | 95.32 |
| TAE | 95.82 | 95.66 | 95.76 | 96.44 |
| ***%LE*** | 3.9 | 0.726316 | 0.084211 | 7.942105 |
| ***%AE*** | 0 | 0 | 0 | 9.657895 |
| ***DR*** | 3.588593 | 4.149321 | 4.352193 | 3.53532 |
| ***MEE*** | -2.30202 | -2.29437 | -2.44726 | -3.49016 |
| ***STDEE*** | 2.322125 | 2.322165 | 2.320881 | 2.167919 |
| ***POWEE*** | 10.69157 | 10.65659 | 11.37558 | 16.88107 |

Furthermore, the time percentage %LE in which a sensor is declared "suspect" *before* a true failure detection is reported to assess the false detection (false alarm) rate. Similarly the time percentage %AE in which a sensor is declared "faulty" *before* a true failure declaration is evaluated to assess the false accommodation rate. Finally, the parameters *MEE(k), STDEE(k)* and *POWEE(k)* are introduced to evaluate the goodness of the accommodation system. These parameters represent the mean, the variance and the power of the absolute estimation error over the whole data file.

Table IV and V show these results for the two architectures. Analysing the results in Table V, some conclusion can be drawn for the MLP approximator. For moderate values of the learning rate a substantial decrease in false alarms is observed. An increase in the detectability ratio DR can be seen without a corresponding increase in the power of the error (POWEE). The price to pay is generally an increase in the delay for failure declaration (TLE, TAE); however, this effect is relatively modest. These results highlight the benefit of introducing an on-line learning scheme, at least for MLP approximators, to improve the degree of reliability of the detection scheme and to guarantee satisfactory performances in all the working conditions. However the learning rate cannot be increased arbitrarily; in fact, an excessive value increases the possibility that the SFDIA scheme “learns” a faulty pattern before a failure is declared and properly accommodated. This risk is confirmed by the analysis of the last columns in the tables, where high learning rates prevented the detection of the failures. This is also consistent with a significant decrease in the detectability ratio. Furthermore higher learning rates compromise the integrity of the global approximation capability of the NN, which can have adverse effects at post-failure conditions when the neural approximator has to supply reliable estimates using only the learned data. The fact that for low learning rates the post-failure POWEE index takes on small values shows the possibility of using the neural estimator as “virtual sensor” after a failure declaration. The RBF-EMRAN NNs have generally shown better performance. Their approximation and generalization capabilities are confirmed by the absence of false alarms even without learning, by the high values of the detectability ratio, and by the lowest values of the POWEE index.

## Fault Tolerance in Flight Management

Management of the aircraft formation can be centralized or decentralized. In the former case, one formation manager exists that acts as a supervisor for all the aircraftand manages the topology of the channels they use to exchange information among themselves. This manager can be one of the aircraft or ground-based. The major advantage of ground-based management is the capability to react to and reach decisions from a possibly higherlevel of "intelligence" than that achievable by on-board computers. In fact, human decision could be added to the automatic system but this surely introduces a delay in the reconfiguration progress. A prompt response is still needed to bring the formation into a safe configurationprior to a ground-based decision. In a decentralized management scheme, on the other hand, each aircraft is given a certain level of decision capability, while the whole formation must be capable of reconfiguring, making decisions, and achieving mission goals. In both cases it is important to establish general rules for the information exchange and graph theory can be used to describe inter-aircraft communication. The aircraft can be thought of as the nodes of a graph. The physical communication channels create the arcs in the graph. These arcs are oriented because, in the most general case, channels are not bi-directional; this is not a limitation, since two nodes can be connected by two opposite direction arcs to model a bi-directional channel. The graph must also be connected because, if two subgraphs exist without any arc connecting them, the aircraft inside the two groups cannot behave like a formation component, yielding in fact two separate formations. The communications graph should be redundant from the standpoint of the capability of propagating information. In the event of failures, this redundancy leaves room for reconfiguration. However, having the capability of using a channel does not mean that it must be used at all times. Optimization of available channels under a cost function constraint can be achieved via graph programming techniques.

### *Graph Theory Optimization*

The problem of mapping a formation structure with graph theory can be configured as a shortest path optimization problem. Suppose that at least one of the aircraft knows the mission reference trajectory. This reference trajectory can be seen as the effective formation leader and can be represented in the graph by a node called Virtual Leader (VL), so that each aircraft that knows the mission trajectory has one communication channel with the VL. If a cycle exists, since the graph must be connected, it must contain all nodes. However because at least one node has the VL as its only reference, and the VL has no incoming arcs by definition, the VL and all nodes using the mission trajectory as reference cannot be part of a cycle. Thus the graph cannot have any cycle, and the VLwill be the root of the solution tree.

To solve the shortest path problem, the Dijkstra algorithm was used. It has polynomial complexity, it guarantees optimality of the solution, and it is deterministic; that is, starting from the same conditions, all runs lead to the same solution. The Dijkstra algorithm can be modified to obtain the optimal redundant channels among those outside theoptimal arcs set. Suppose we need *m* total possible channels for each node; that is, *m-1* redundant channels. The unmodified algorithm can still be used to find the optimal solution. At the end of the optimization procedure, the nodes' potentials are frozen. Then for all nodes *i* in the S set, select *m-1* incoming arcs that have the minimum value of $d'(i) = d(j)+Cji$, where *j* is a possible redundant preceding node, and order for increasing values of $d'(i)$. This modification computes sub-optimal solutions. The arcs chosen with this technique can be considered suboptimal because they are the second-best choices. After removing the incoming arc of a node belonging to the optimal path, the new optimal incoming arc that a new run of the Dijkstra algorithm will produce is the first one of this redundant arcs list. Figure 22 shows the graph of a sample formation. While the VL is connected with all aircraft, not all possible inter-aircraft connections are present. An arc's weight is identified by a positive number. The figure also shows the result of the modified Dijkstra algorithm. The optimal path, in solid line, and each node. In this case, *m=3* but a node might not have the two required redundant channels because it has fewer than m incoming arcs; this is the case for node 1. When node A switches reference and uses a sub-optimal path, its potential increases, and a reconfiguration procedure must be run on all the nodes of the sub-tree that originates from that node. In fact, a new global optimization would be needed, but the potential of all the nodes that do not have any incoming optimal path from A cannot be affected, because these surely have a lower cost path to the VL. Since the detected failure and the new node A potential after reconfiguration must be propagated to the nodes that belong to the optimal A sub-tree path only, the same communications channels used for formation-keeping data can also be used to exchange potential updates. Furthermore, since after all nodes have completed the reconfiguration, the new graph is optimal, this procedure can be repeated in case of successive faults without ever having to reconsider optimization of the whole graph. This means that any number or combination of successive faults can be managed with this sub-tree-based technique without compromising whole-graph optimality.

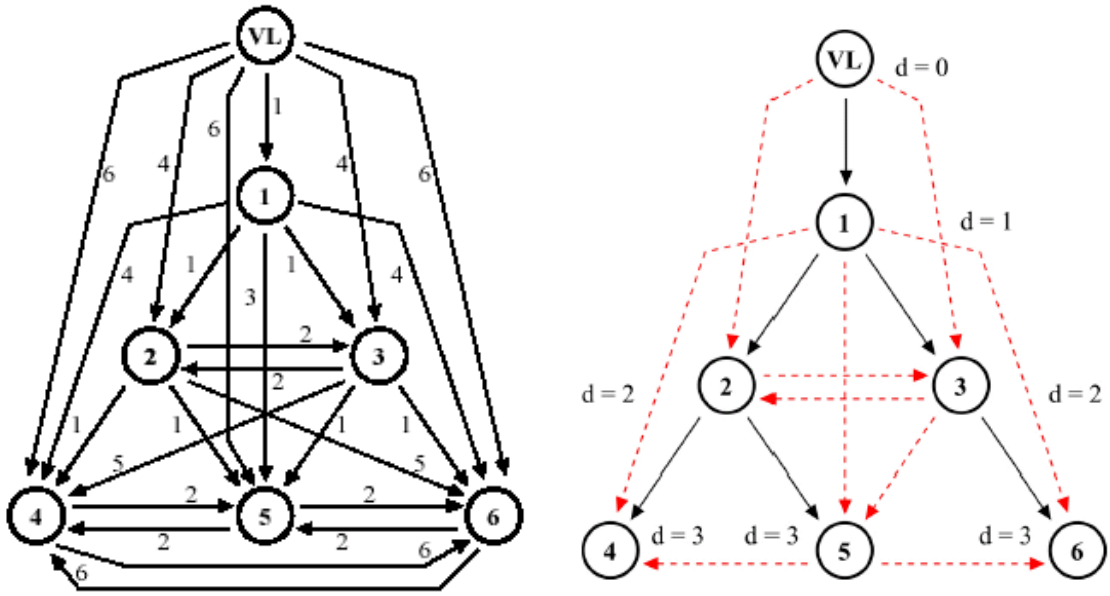


Figure 22. Fault Tolerant Communications Link for Formation Flight

The following simulation presents the reconfiguration of the formation, shown in figure 22, following a fault in the aircraft 3 transmitter. The failure happens during a 180-deg turning maneuver. As shown in the figure, aircraft 6 uses aircraft 3 as the reference for its formation-keeping control system. The first redundant channel for aircraft 6 is that coming from aircraft 5. When aircraft 6 detects a fault, it reconfigures itself to use aircraft 5 as reference. Figure 23 shows the distance of aircraft 6 from its ideal trajectory compared to that of the same simulation without failure. The red line shows that, after the fault at = *50* sec, a fast transient is followed by a reconfiguration that brings the aircraft back into a stable position inside the formation. The two traces of distance error do not have to join, in this case, because the performance of the controller is different.

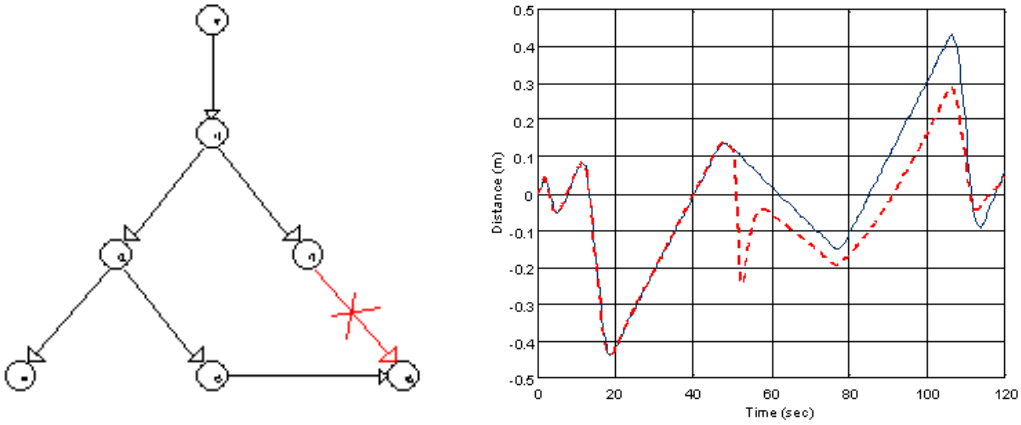


Figure 23. Failure Reconfiguration during Formation Flight

## Conclusions

This paper has presented some results of a study undertaken with the goal of showing the potential of online learning neural networks for adaptive control at non-linear conditions. Particularly the paper has shown successful simulations of identification and (identification + control) using an indirect approach for non-linear SISO systems. The paper presented also some issues related to the use of neural architectures in the area of sensor and actuator failure detection and identification. The main focus of the work was the problem of achieving satisfactory performance for on-line learning, and subsequent potential for on-board implementation on flight vehicles having low and/or no physical redundancy. To this end standard gradient-based neural networks were considered and adapted to the above constraints. A NN-based scheme SFDIA has been analyzed using experimental flight data from the WVU B777 research aircraft model. T he scheme was implemented with 2 neural architectures, that is the MLP-EBPA NN and the RBF-EMRAN NN. The scheme has shown to be successful in the detection, isolation and accommodation of failures “injected” on the WVU B777 flight data. Finally some issues on failure accomodation in formation flight management have been addressed using linear programming techniques.

## Acknowledgement

This work was supported in part grants from the NASA West Virginia Space Consortium, the Institute for Software Research, and the Italian Ministry of Education and University Research.

## References

1. Astrom, K.J, Wittenmark, 1989, "*Adaptive Control*", Addison-Wesley Publishing Co.
2. Barnard, E.,1992 "Optimization for Training Neural Nets", *IEEE Transactions on Neural Networks*, Volume 3, Number 3, pg. 232
3. Casdorph, V.A., 1994 "Detection and Identification of Battle Damage and/or Generic Failure Using On-Line Learning Neural Networks and Cross Correlation Functions", Master Thesis, West Virginia University, USA.
4. Chen, C.L., Nutter, R.S., 1992 "An Extended Back-Propagation Learning Algorithm by Using Heterogeneous Processing Units", IJCNN92, Baltimore, Md, USA.
5. Chen, V.C., Pao, Y.H., 1989, "Learning Control with Neural Networks", Proceedings of the International Conference on Robotics and Automation.
6. Cybenko, G. , 1989, "Approximation by Superposition of Sigmoidal Functions", *Mathematics of Control Signals and Systems*, Vol.2, No.4, pg. 303.
7. De Villiers,J., Barnard, E., 1993, "Back-Propagation Neural Nets with One and Two Hidden Layers", *IEEE Transactions on Neural Networks*, Volume 4, Number 1, pg. 136.
8. Freund, E., 1975, "The Structure of Decoupled Non-Linear Systems", *International Journal of Control*, Vol. 21, pg.651.

9. Hornik, K., Stinchcomb, M., and White, H., 1989, "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks*, Vol.2, pg.359.
10. Hunt, K.J., Sbarbaro, D., Zbikowski, R., and Gawtrhop, P.J., 1992 "Neural Networks for Control Systems - A Survey", *Automatica*, Vol. 28, No. 6, pg.1083.
11. Isidori, et al., A., 1981, "Non-Linear Decoupling Via Feedback : a Differential Geometric Approach", *IEEE Transactions on Automatic Controls*, Vol. AC-26, pg.341.
12. Kawato, M., Furukawa, F., and Suzuki, R., 1987, "A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement", *Biological Cybernetics*, Vol.57, pg.169.
13. Napolitano, M.R., Chen, C.I., and Naylor, S., 1993, "Aircraft Failure Detection and Identification Using Neural Networks", *AIAA Journal of Guidance, Control and Dynamics*, Volume 16, Number 6, pg. 999.
14. Narendra, K.S., Monopoli, R.V., 1980 "*Applications of Adaptive Control*", New York Academic Press.
15. Narendra, K.S., Partasarathy, K., 1990, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, Vol.1, No.1.
16. Naylor, S.M., 1994, "On-Line Learning Non-Linear Neural Controllers for Restructurable Flight Control Systems", Master Thesis, West Virginia University, Morgantown, WV, USA.
18. Neppach, C.D., 1994, "Application of Neural Networks in Sensor Failure Detection, Identification and Accommodation in a System Without Sensor Redundancy", Master Thesis, West Virginia University, Morgantown, WV, USA.
19. Nielsen, R.H., 1991, "*Neurocomputing*", Addison Wesley Publishing Company.
20. Ogata, K., 1987, "*Discrete-Time Control Systems*", Prentice-Hall, Inc.
21. Psaltis, D., Sideris, A., and Yamamura, A., 1988, "A Multilayered Neural Network Controller", *IEEE Control Systems Magazine*, Vol. 4, pg.17.
22. Sastry, S.S., Isidori, A., 1989 "Adaptive Control of Linearizable Systems", *IEEE Transactions on Automatic Controls*, Vol. 34, No.11, pg.1123.
23. Simpson, P.K., 1990, "*Artificial Neural Systems*", Pergamon Press Inc., Fairview Park, NY.
24. Singh, S.N., and Rugh, W.J., 1972, "Decoupling in a Class of Non-Linear Systems by State Variable Feedback", *ASME Transactions*, Vol. 94, pg.323.
25. Rumelhart, D. and McClelland, J., 1986, "*Parallel Distributed Processing*", MIT Press, Cambridge, MA,USA.
26. Venugopal, K.P., et al., 1994 "An Improved Scheme for Direct Adaptive Control of Dynamical Systems Using Backpropagation Neural Networks", accepted for publication to be published in "*Journal of Circuits, Systems and Signal Processing".*
27. Widrow, B., and Stearns, S.D., 1985, "*Adaptive Signal Processing*", Prentice Hall, Inc.
28. NASA, 1993,"Cooperative Agreement Award No. NCCW-0051", with West Virginia University.
ASI, 1995-1997, "Sviluppo di Architetture Neurali Software e Hardware per la Identificazione in Linea di Guasti in Sensori e Attuatori di Veicoli Aerospaziali", Contratto di Ricerca Agenzia Spaziale Italiana-Consorzio Pisa Ricerche.
30. Willsky, A.S., 1976, "A Survey of several Failure Detection Methods", *Automatica*, Vol. 12, No. 6.
31. Guo, T., Nurre, J., 1991, "Sensor Failure Detection Recovery using Neural Networks", *Proc. of the IJCNN '91*.
32. Napolitano, M.R., Innocenti, M., et. al, 1994, "Sensor Failure Detection, Identification, and Accomodation using a Neural Network based Approach", *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Scottsdale, AZ, August.
33. Silvestri, G., 1995, "Implementazione real-time di reti neurali in applicazioni di identificazione e di controllo in presenza di malfunzionamento sui sensori", Ph.D. dissertation, University of Pisa, Italy.
34. Napolitano, M.R., Innocenti, M., et. al., 1995, "Neural-Network-Based Scheme for Sensor Failure Detection, Identification, and Accomodation", *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6.
35. Napolitano, M.R., Casdorph, V., Neppach, C., Naylor, S., Innocenti, M., Silvestri, G, 1996, "Online Learning Neural Architectures and Cross-Correlation Analysis for Acutator Failure Detection, and Identification", *International Journal of Control*, January.
36. Napolitano, M.R., Casdorph, V., Neppach, C., Naylor, S., Innocenti, M., Silvestri, G, 1994, "Implementation of Hardware-Based On-Line Neural Architectures for Sensor Failure Detection, Identification, and Accomodation", submitted for publication to the *AIAA Journal of Guidance, Control, and Dynamics*, October.

37. Napolitano, M.R.m Windon, D., Casanova, J., Innocenti, M., 1996, "A Comparison between Kalman Filter and Neural Network Approaches for Sensor Validation", AIAA-96-3894, Guidance and Control Conference, San Diego, California, August.
38. Del Gobbo, D., 1996, "Tecniche avanzate per la Rilevazione di Malfunzionamenti", Master thesis, Universita' di Pisa, Italy.
39. Patton R.J., Frank P.M., Clark RN. "*Fault diagnosis in dynamic systems: Theory and applications",* Englewood Cliff, NJ, Prentice-Hall, 1989.
40. Baruh H., Choe K. "Sensor-Failure Detection Method for Flexible Structures", *AIAA Journal of Guidance, Control, and Dynamic 1987*; Vol. 10, no 5, 474-482.
41. Kerr T.H. "False Alarm and Correct Detection Probabilities over a Time Interval for Restricted Classes of Failure Detection Algorithms", *IEEE Transactions of Information Theory 1982*; IT-28, No. 4, pp. 619-631.
42. Napolitano M.R., Neppach C.D., Casdorph V., Naylor S., Innocenti M., Silvestri G. "A Neural Network Based Scheme for Sensor Failure Detection, Identification and Accommodation", *AIAA Journal of Guidance Control and Dynamics 1995;* 18(6) 1280-1286.
43. Hunt K.J., Sbardato D., Zbikowski R., Gawthrop P.J. "Neural Networks for Control Systems: a Survey" *Automatica 1992;* 28(6), pp. 1083-1112.
44. Napolitano M.R., An Y., Seanor B. "A Fault Tolerant Flight Control System for Sensor and Actuator Failure using Neural Networks", *Aircraft Design 2000;* Vol. 3, pp. 103-128.
45. Vemuri A., Polycarpou M., Diakourtis S. "Neural Network Based Fault Detection and Accommodation in Robotic Manipulators", *IEEE Transaction on Robotics and Automation* 1998; Vol. 14, no. 2, pp. 342-348.
Napolitano M.R. "*Design of the B777 Model Flight Control System"*. WVU Internal Technical Report, March 1999.
46. Isermann R., Balle' P. "Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes", *Control Engineering Practice 1997;* Vol. 5, no. 5, pp. 709-719.
47. Lu Y., Sundararajan N., Saratchandran P. "Analysis of Minimal Radial Basis Function Network Algorithm for Real-Time Identification of Non-linear Dynamic Systems". *IEEE Proceedings on Control Theory and Application 2000;* Vol. 4, no. 147, pp. 476.